E-coli Propagation: Monte Carlo Simulation

University of California, Berkeley, Department of Chemistry
Masters of Molecular Science and Software Engineering (MSSE)

Team:	Brendan Dang, Manula Dombagahawatta, Jinu Jung, Carlos Lopez, Jade Warren
Class :	CHEM273
Program:	MSSE
Assignment:	Project 2
Primary Instructor:	Dr. Markus Hohle

Introduction: This project simulates **E.coli** behavior in two dimensions. E.coli, like many bacteria, exhibits chemotaxis, a cell behavior where the detection of some kind of extracellular signal gradient allows for directional movement of the cell.[1]. This kind of behavior can be also be described by and modeled using "run-and-tumble" dynamics.[2] In this project, we model E.coli movement toward a food source by alternating between random movement (the "tumble") and directed movement (the "run") along a specific axis which is determined by information gained during the previous tumble phase. This resulted in an implementation of an m-order Markov chain where we defaulted to storing $m = 4$ time steps in the chain memory.

1 Project Methodology

1.1 Approach

There were a few ways to approach this simulation. We knew that we must support random movement in both the x and y directions, and we know that we must instantiate some kind of concentration gradient that the moving object can read. We chose to characterize the nutrient concentration as continuous and then moved our E-coli in discrete amounts dx and dy in the x and y directions independent of a set grid that depended on the phase, step-size, and learning rate of the cell. In other words, we made sure that the amount of nutrients at any position, (x, y) could be described by some kind of continuous function, $C(\mathbf{x},\ \mathbf{y})$.

1.2 Profile Examples

When we first started, we aimed to explore two basic function options, namely:

- **Gaussian:** where $C_{\text{gauss}}(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{(x-\mu_x)^2}{2\sigma_x^2} - \frac{(y-\mu_y)^2}{2\sigma_y^2}\right)$ and σ_x and σ_y are the standard deviations in the x and y directions.
- **Exponential:** $C_{\text{exp}}(\mathbf{r}) = A \exp\left(-\frac{\|\mathbf{r}-\mu\|}{\lambda}\right)$ where A is the peak concentration at the center μ and λ controls the decay rate.

where those functions were characterized by way of a probability density function. However, when we started looking at additional functional support, such as for the log-logistic functions (where, for example, in the univariate case, $f(x, c) = \frac{cx^{c-1}}{(1+x^c)^2}$), we began looking at, rather, than using equations as given above, popular probability density functions.

1.3 Making It Dynamic

We then chose to make the value (and consequently, any differences between values at different points) of the concentration at any point (x, y) equivalent to some constant S (the strength of the concentration) times the probability of being at at that position on the grid. The spread of the concentration was described by the distribution variance, and the mean was described with some random set of values (x_i, y_i) that were randomly generated in the middle half of the initial positional grid. In other words, $C(x, y)$ became:

$$S \times P(x = x_i, y = y_i)$$

and concentration gradients were best described by:

$$\Delta C = S \times (P(x = x_{current-4\Delta t}) - P(x = x_{current}))$$

where P is the relevant probability density function.

2 Implementation

2.1 Tumble Phase

In the tumble phase, each bacterium moves randomly for a small number of steps (default number of iterations = 4).
Here is the python code:

```
def __tumble(self, x, y, time_step, num_iterations=4, step_size=2)
    dx = np.zeros((num_iterations, *x[time_step, :].shape))
    dy = np.zeros((num_iterations, *y[time_step, :].shape))
    dx = step_size * np.random.uniform(-1, 1, (num_iterations, *x[time_step, :].shape))
    dy = step_size * np.random.uniform(-1, 1, (num_iterations, *y[time_step, :].shape))
    new_x = np.cumsum(dx, axis=0)
    new_y = np.cumsum(dy, axis=0)
    self.tumble_sizes.append(np.sqrt(new_x[-1, :]**2 + new_y[-1, :]**2))
    new_x += x[time_step - 1, :]
    new_y += y[time_step - 1, :]
    x[time_step : time_step + num_iterations, :] = np.clip(new_x, self.x_min, self.x_max)
    y[time_step : time_step + num_iterations, :] = np.clip(new_y, self.y_min, self.y_max)
    return x, y
```

This function makes the bacteria "tumble" by taking a few random steps in both x and y directions while keeping them within the allowed range using the np.clip function. The random values in the range -1 to 1 for both dx and dy allow the bacterium to move forward or backward, and the step size input scales the movement. A cumulative sum along the tumble steps then finds the displacement from the starting point. The new coordinates are stored and the method returns the movement history by way of position organized by time step.

2.2 Run Phase

In the run phase, movement is directed by the nutrient gradient. We can find that using the previously stored movement history.
Here is the python code:

```
def __run(self, x, y, time_step, step_size: int=3, delta: int=4):
    delta_x = x[time_step - delta - 1, :]
    prev_x = x[time_step - 1, :]
    delta_y = y[time_step - delta - 1, :]
    prev_y = y[time_step - 1, :
    delta_x_y = np.stack((delta_x, prev_y), axis=1)
    prev_x_y = np.stack((prev_x, prev_y), axis=1)
    dz_dx_dt = self.strength * step_size * ( self.conc_dist(prev_x_y) - self.conc_dist(
        delta_x_y)) / delta * np.sign(prev_x_y[:, 0]-delta_x_y[:,0])
    delta_y_x = np.stack((prev_x, delta_y), axis=1)
    prev_y_x = np.stack((prev_x, prev_y), axis=1)
    dz_dy_dt = self.strength * step_size * (self.conc_dist(prev_y_x) - self.conc_dist(
        delta_y_x)) / delta * np.sign(prev_y_x[:, 1] - delta_y_x[:,1])
    self.run_sizes.append(np.sqrt(dz_dx_dt**2 + dz_dy_dt**2))
    x[time_step, :] = prev_x + dz_dx_dt * self.learning_rate
    y[time_step, :] = prev_y + dz_dy_dt * self.learning_rate
    return x, y
```

There is a larger step size than was defaulted to in the tumble method, and the movement delta $< dx, dy >$ is scaled by the learning rate of the simulation, the magnitude of the gradient, and the step-size. As it is guided movement, note that the direction of the new movement is given by the change in gradient over time multiplied by the sign of the change in position. This is done discretely in both the x and y directions. Much like the previous class method for tumbling, the method returns the updated movement history after the "run" phase.

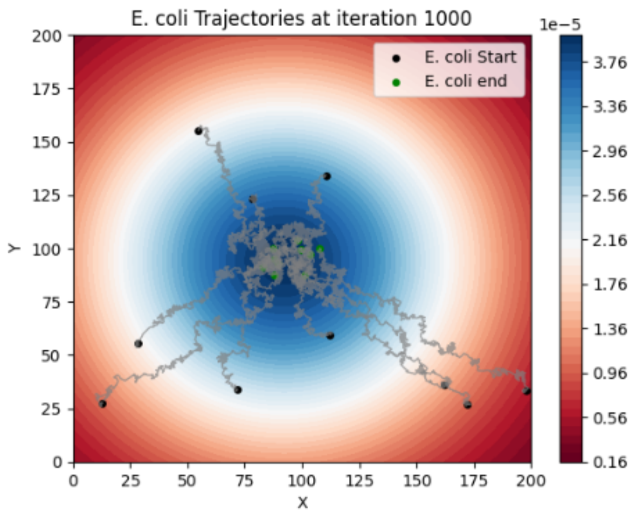

Figure 1: Concentration Distribution Profile = Gaussian

2.3 Histogram

To see how the population moves toward the nutrient peak over time, we plot histograms showing each cell's distance from the food source at several key iterations. This was simplest for the normal distribution, as the concentration center stored as a value specific to an instance of the simulation class is very clearly the point at which the E.coli should converge. For other distributions like the exponential, where the maximum ends up very close to the bounds of the simulation, and bimodal distributions, the histogram results for large N number of E.coli can be quite varied.
The below is code from the original implementation with a Gaussian distribution in mind. Each snapshot is shown as a histogram with 200 bins and a distance $D = \sqrt{x_{\max}^2 + y_{\max}^2}$,ensuring the results are directly comparable across different times.
Here is the python code:

```
def get_histogram(self, ecoli_x_pos, ecoli_y_pos):
       time_stamps = [1, 10, 100, 999]
       distances_from_food = np.zeros((len(time_stamps), self.num_ecoli))
       for i in range (len(time_stamps)):
           x_distances_from_food = ecoli_x_pos[time_stamps[i]-1] - self.con_center[0]
           y_distances_from_food = ecoli_y_pos[time_stamps[i]-1] - self.con_center[1]
           distances_from_food[i, :] = np.sqrt(x_distances_from_food**2 +
               y_distances_from_food**2)
       num_bins = 200
       ...
```

3 Project Results

It supports three different distributions- Gaussian, Fisk, and Exponential. In each environments, E.coli alternates between tumbling and running.

3.1 Simulation Images

The following images show the simulated movement of E.coli in Gaussian. In the simulation, the bacteria alternate between tumbling(random changes in direction - dx and dy) and running (moving toward high concentration). Black dots mark the starting positions, and green dots mark the final positions, and gray lines trace the movement paths over time.

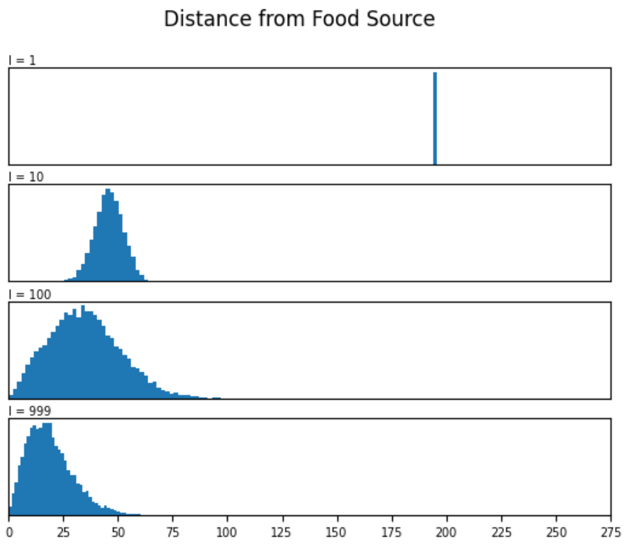

Figure 2: Concentration Distribution = normal, N = 10000, n_{bins} = 200

3.2 Histogram Results

The histograms show the distribution of distances from the Food Source. In the histograms, i = 1,20,100, and 999 refer to specific iterations in the simulation. These images show the distribution of distances from the food source at different stages(e.g i = 1 is starting point, i = 999 is ending point). Histogram exhibits how the E.coli population moves closer to the nutrient center over time.

4 Discussion

The simulation results demonstrate that E.coli populations in three concentration profiles(Gaussian, Fisk, and Exponential) with different N values(10,100,1000).
The gaussian environment showed the most consistent and smooth convergence, because its gradient is symmetric and continuous. In contrast, the Fisk and Exponential profiles resulted in more variable movement patterns, due to uneven nutrient distributions influencing the paths taken.

5 Project Limitations

Simulation only observes E.coli behavior under idealized and simplified assumptions, rather than fully reproducing real-world conditions. We encountered many edge cases in managing gradient magnitudes, which was tackled through a combination of dampening and clipping gradient values. If we were to consider the E.coli behavior more carefully, there may be a more sophisticated method to calculate the run distances.

5.1 Simplified Movement

The movement of E.coli is modeled only as alternating between ”tumble” and ”run”. It does not assume any complex variations in speed and turning behavior. We assumed that E.coli have same speed and concentration gradients.

5.2 Environments

Simulation is limited to a Two Dimensional grid, but the real world E.coli is 3 Dimensions. In addition, The environment is static, the food source does not get reduced, and factors like temperatures are not considered.

References

[1] Jin T., Hereld, D. (2006). Moving toward understanding eukaryotic chemotaxis. *European Journal of Cell Biology*, *85*(9-10), 905-913. https://doi.org/10.1016/j.ejcb.2006.04.008.

[2] Kurzthaler, C., Zhao, Y., Zhou, N., Schwarz-Linek, J., Devailly, C., Arlt, J., Huang, J., Poon, W., Franosch, T, Tailleur, J. Martinez, V. (2024). Characterization and Control of the Run-and-Tumble Dynamics of Escherichia Coli. *Phys. Rev. Lett.*, *132*(3), https://link.aps.org/doi/10.1103/PhysRevLett.132.038302.